Fevzi KILAS - 2200356822 - BBM 409 Report

● Custom Design CNN Model:

The model starts with three convolutional layers to extract hierarchical features from the input images. Each convolutional layer consists of a 2D convolution operation followed by a ReLU activation function to introduce non-linearity and enhance feature representation. The number of filters increases progressively to capture more complex patterns and features. Max pooling layers are inserted after each convolutional layer to downsample the feature maps, reducing spatial dimensions.

- Convolutional Layer 1:

  Input: RGB image with a size of 224x224x3
  Number of filters: 32
  Filter size: 3x3
  Stride: 1
  Padding: 1
  Activation: ReLU
  Max Pooling: 2x2

-Convolutional Layer 2:

  Number of filters: 64
  Filter size: 3x3
  Stride: 1
  Padding: 1
  Activation: ReLU
  Max Pooling: 2x2

-Convolutional Layer 3:

  Number of filters: 128
  Filter size: 3x3
  Stride: 1
  Padding: 1
  Activation: ReLU
  Max Pooling: 2x2

After the convolutional layers, the feature maps are flattened into a 1D vector and passed through two fully connected layers. The first fully connected layer contains 128 neurons with ReLU activation, allowing for higher-level feature extraction and nonlinear transformations. Dropout regularization is applied to mitigate overfitting by randomly zeroing out a fraction of the neurons during training. The final fully connected layer outputs the classification probabilities for the two skin lesion classes (benign and malignant) using the softmax activation function.

-Fully Connected Layer 1:

Input size: 128 x 28 x 28 (resulting feature map size from the last convolutional layer)
Number of neurons: 512
Activation: ReLU
Dropout: 0.5 (to reduce overfitting)

-Fully Connected Layer 2 (Output layer):

Number of neurons: 2 (for binary classification - benign or malignant)
Activation: None (output logits)

The output layer consists of two neurons, corresponding to the two skin lesion classes. It produces the final classification predictions.

The model contains learnable parameters, including the weights and biases associated with the convolutional and fully connected layers. These parameters are adjusted during training to minimize the loss function and improve the model's performance.

The diagram below illustrates the flow of data through the custom CNN model:

Input (224x224x3) -> Conv1 (32 filters, 3x3) -> ReLU -> Max Pooling (2x2) -> Conv2 (64 filters, 3x3) -> ReLU -> Max Pooling (2x2) -> Conv3 (128 filters, 3x3) -> ReLU -> Max Pooling (2x2) -> Flatten -> FC1 (512 neurons) -> ReLU -> Dropout (0.5) -> FC2 (2 neurons - output logits)

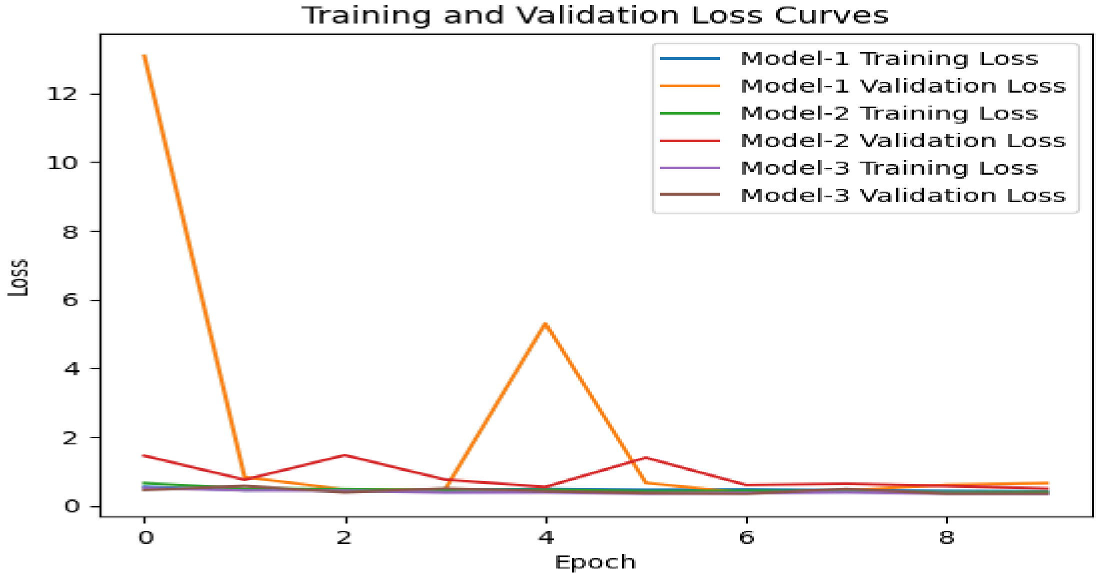
The input to the model is an RGB image with a size of 224x224x3. The image passes through three convolutional layers, each followed by a ReLU activation function and max pooling to downsample the feature maps. This hierarchical pattern of convolutional and pooling layers helps the model learn hierarchical features at different scales.
After the convolutional layers, the output feature maps are flattened and passed through two fully connected layers. The first fully connected layer reduces the dimensionality of the feature maps and applies a ReLU activation function. To prevent overfitting, a dropout layer with a probability of 0.5 is applied before the second fully connected layer, which produces the final logits.

The model uses the Cross-Entropy Loss function for training and Adam optimizer for optimization.

In this architecture, the convolutional layers perform local feature extraction, capturing patterns such as edges, textures, and shapes. The pooling layers downsample the feature maps, reducing the spatial dimensions and retaining the most salient features. The fully connected layers at the end of the model combine the learned features and make the final classification predictions.

● Test Results & Visualizations:

For Step 1:



model 1: Test Accuracy: 0.8047, Confusion Matrix: [[293 11] [57 18]]

```
Classification Report:
          precision    recall  f1-score   support
       0       0.84      0.96      0.90       304
       1       0.62      0.24      0.35        75

  accuracy                          0.82       379
 macro avg       0.73      0.60      0.62       379
weighted avg       0.79      0.82      0.79       379
```

model 2: Test Accuracy: 0.8100 Confusion Matrix: [[297   7] [65 10]]

```
Classification Report:
          precision    recall  f1-score   support
       0       0.82      0.98      0.89       304
       1       0.59      0.13      0.22        75
  accuracy                          0.81       379
 macro avg       0.70      0.56      0.55       379
weighted avg       0.77      0.81      0.76       379
```

model 3: Test Accuracy: 0.8364   Confusion Matrix: [[299   5] [65 10]]

```
Classification Report:
          precision    recall  f1-score   support
       0       0.82      0.98      0.90       304
       1       0.67      0.13      0.22        75
  accuracy                          0.82       379
 macro avg       0.74      0.56      0.56       379
weighted avg       0.79      0.82      0.76       379
```

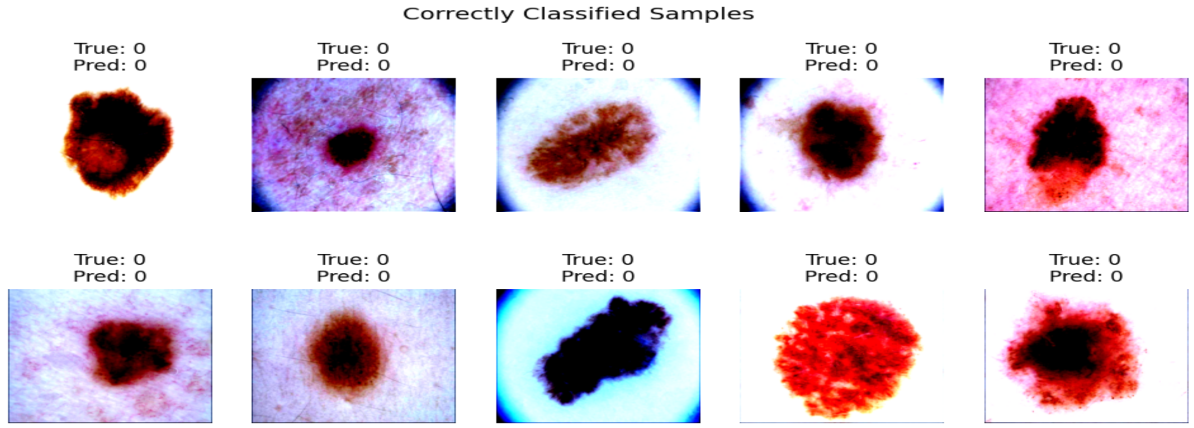
For Step 2:

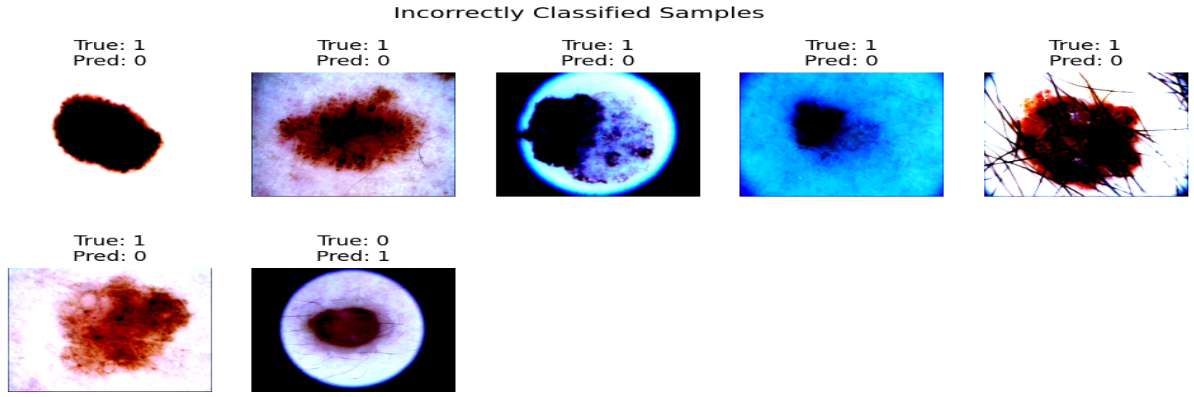
CNN Model:

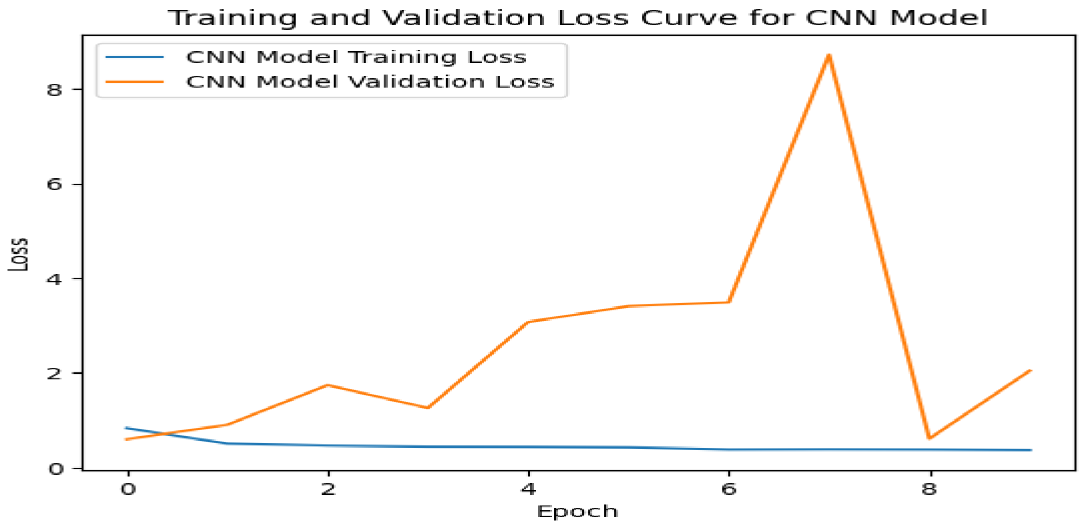
Test Accuracy: 0.8047 Confusion Matrix: [[297   7] [69   6]]

Classification Report:

```
           precision    recall  f1-score   support
        0       0.81      0.98      0.89       304
        1       0.46      0.08      0.14        75
    accuracy                        0.80       379
   macro avg       0.64      0.53      0.51       379
weighted avg       0.74      0.80      0.74       379
```






Training and Validation Loss Curve for CNN Model
CNN Model Training Loss
CNN Model Validation Loss
Loss
Epoch